PENTAHO BEST PRACTICES

Embedding using iFrames

Version 1.0

Gianluca Natali

Table of Contents

Overview

This document is intended to provide best practices for integrating Pentaho inside another application via iFrames. It is not intended to demonstrate how to implement each best practice or provide templates based on the best practices defined within the document.

Software	Version
Pentaho BA Server	8.0

In this document, we'll guide you in the code needed to integrate Pentaho into your application via iFrame. Any code showed in this guide is not to be considered production-ready by any means, and **not supported**. It's only meant for educational purposes.

Embedding Strategies

You can integrate Pentaho in multiple ways:

- Using REST APIs
- Using Pentaho Java Library in your Java application
- Embedding CDE and CDF via requires
- **Via iFrame integration leveraging our HTTP API**

In this document, we'll focus on the latest approach listed, due to its easy implementation and the flexibility to leverage also the **self-service** user experience offered by Pentaho Enterprise edition. You will find more resources to explore also the other approaches in the **References** chapter.

we'll assume that you have a web application where you want to integrate Pentaho.

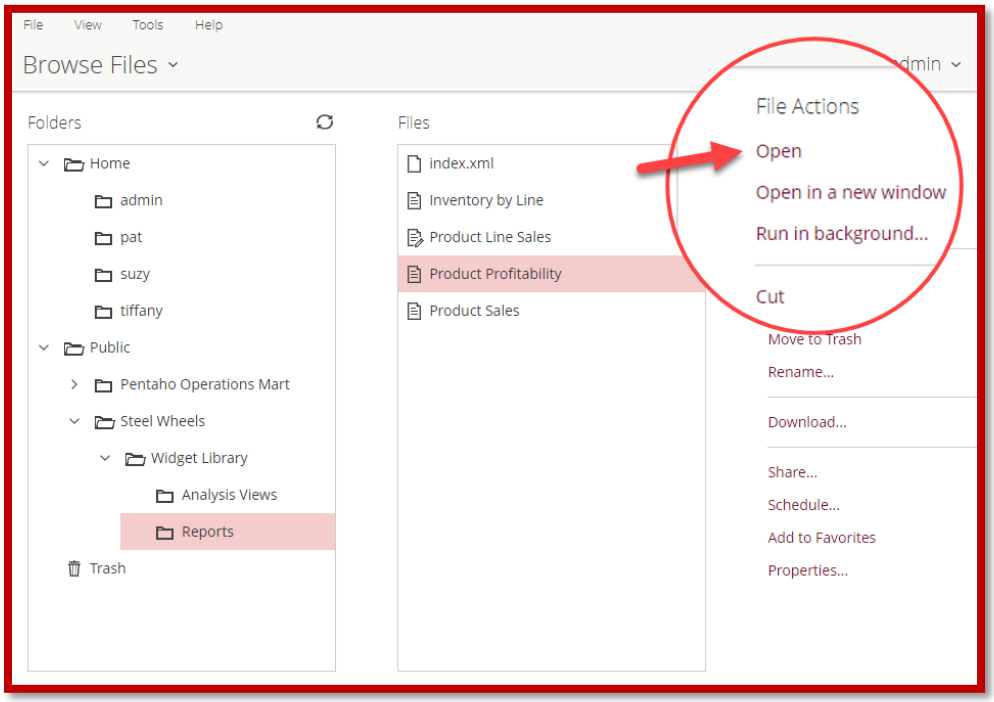
Embedding an existing resource

Depending on the resource you would like to integrate you can use a different URL.

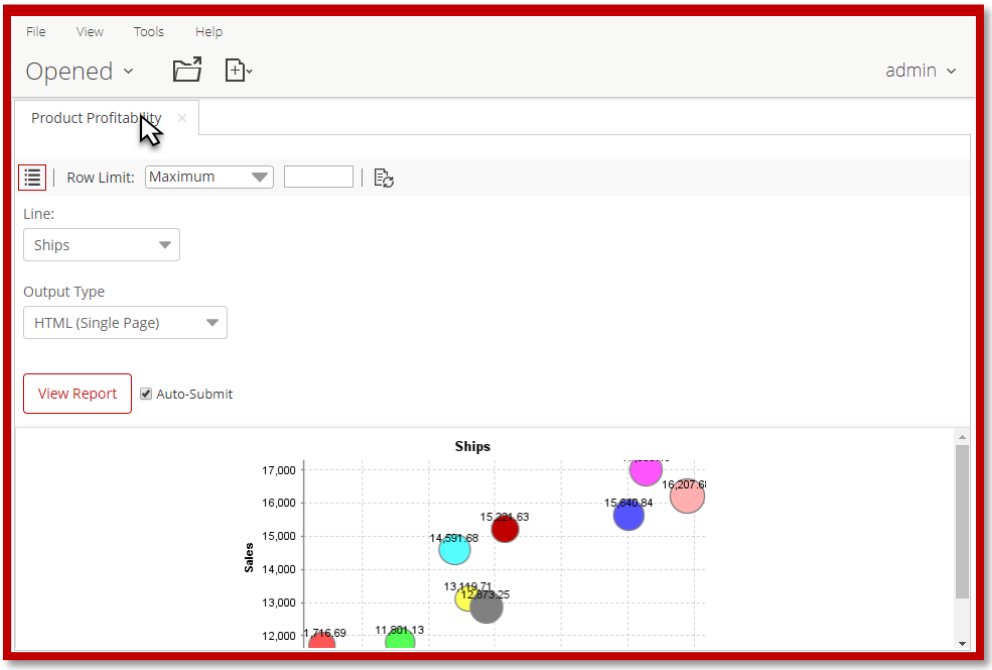
The Simple Way

One of the most simple and known ways to embed a resource from Pentaho is to:

1. Login in the Pentaho User Console
2. Go to Browse Files
3. Find the resource you want to integrate
4. Click on Open (or double-click the resource in the list)



5. The resource will be opened inside Pentaho User Console as a tab
6. Double click the tab where the resource name is



7. In the window that opens copy the URL
8. You can use this URL inside an iFrame

This approach is easy and good for first steps. Anyway, the purpose of this document is to guide you in understanding how that link is generated in the first place, what are the elements that compose it, and how you can write it yourself.

For sake of simplicity, we'll call this HTTP API.

HTTP API

As we mentioned in the previous section, to have the maximum flexibility (that is needed if you are building a Helper as we discuss in the section **Use a helper javascript**) you need a more robust approach. In this section we'll explain how you can build an URL that returns you exactly the feature/element that you want, knowing what each part of the URL means.

Add your Pentaho BA server URL

Please note that for each of the following URL we are omitting the first part, that usually looks like this in an evaluation environment:

http://localhost:8080/pentaho

Pass parameters

You can pass parameters to all these APIs by adding a question mark at the end of the URL, followed by the parameters. This could be used to specify a theme for the UI or to pass parameters to the report/dashboard that you are executing. As example

/api/repos/<path>/viewer?<parameterName>=<parameterValue>

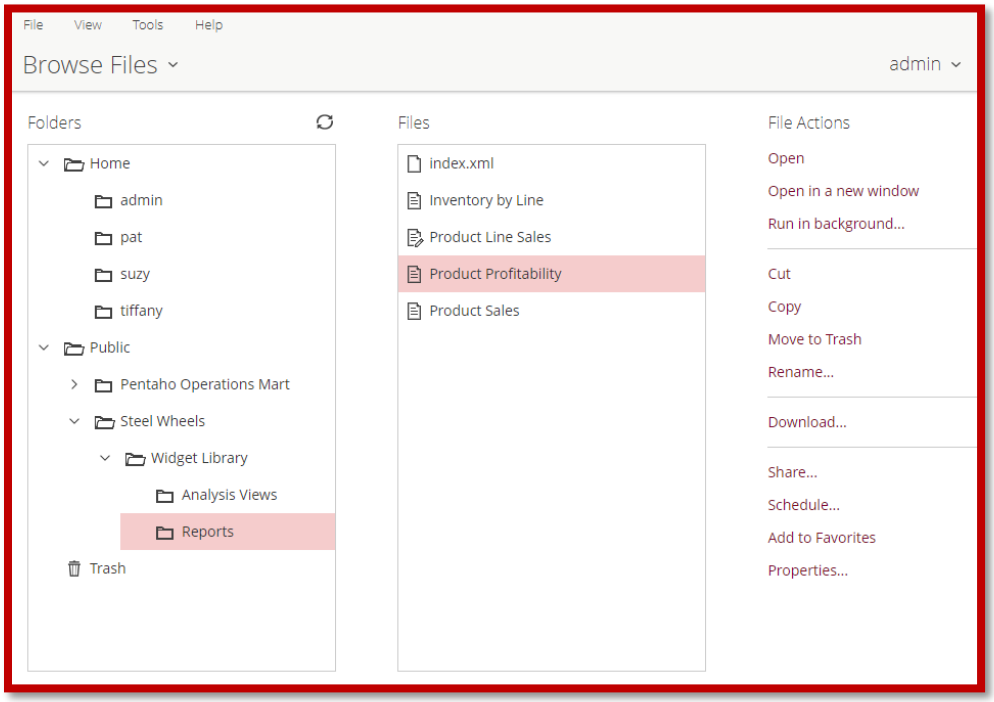
Enable authentication via URL parameters

In a production environment, you would probably be looking at some SSO mechanism to seamlessly integrate the 2 applications without the need for the user to log in.
For an evaluation scenario, a simple username and password in the URL may be a faster solution. You can enable this in Pentaho server as explained in the documentation at this link:

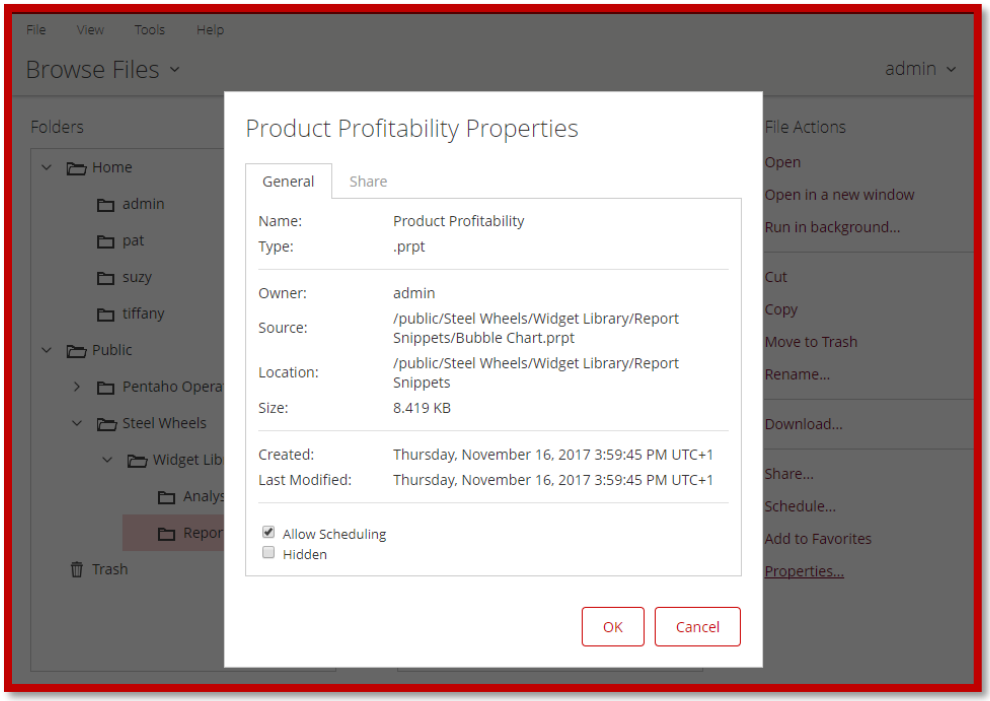
https://help.pentaho.com/Documentation/8.0/Setup/Administration/User_Security/Pass_Authentication_Credentials_in_URL_Parameters

Specify the correct path to your resource

In the following URL examples, you will find a *<path>* placemark. You will have to replace this with the right path for the resource you want to integrate. You can find the path for your resource from the Pentaho User Console, by clicking on it from the Browse Files menu.



Then click on the link Properties… on the right.



You will have to replace the **/** inside the path with a semicolon (**:**).
As an example the path:

/public/Steel Wheels/Widget Library/Report Snippets/Bubble Chart.prpt

Should be written as:

:public:Steel Wheels:Widget Library:Report Snippets:Bubble Chart.prpt

Some characters cannot be included in a URL, and they will have to be encoded. This is usually done automatically by the browser (As an example a space char " " will become "*%20*" . Some other chars must be encoded as the browser will not do that automatically otherwise you will get an error in loading the page.
To learn how to encode URLs have a look at section **Encoding parameters**.

Report

View

```
/api/repos/<path>/viewer
```

Example

```
http://localhost:8080/pentaho/api/repos/:public:Steel Wheels:Widget Library:Report Snippets:Bubble Chart.prpt/viewer
```

Interactive Report

View

```
/api/repos/<path>/prpti.view
```

Edit

```
/api/repos/<path>/prpti.edit
```

Example

```
http://localhost:8080/pentaho/api/repos/:public:Steel Wheels:Vendor Sales Report (interactive report).prpti/prpti.view
```

Analysis Report (Analyzer)

View

```
/api/repos/<path>/viewer
```

Edit

```
/api/repos/<path>/editor
```

Example

http://localhost:8080/pentaho/api/repos/:public:Steel Wheels:Sales Trend (multi-chart).xanalyzer/viewer

Dashboard

View

/api/repos/<path>/viewer

Edit

/api/repos/<path>/editor

Example

http://localhost:8080/pentaho/api/repos/:public:Steel Wheels:Sales Performance (dashboard).xdash/viewer

CDE Dashboard

View

```
/api/repos/<path>/generatedContent
```

Edit

```
/api/repos/<path>/wcdf.edit
```

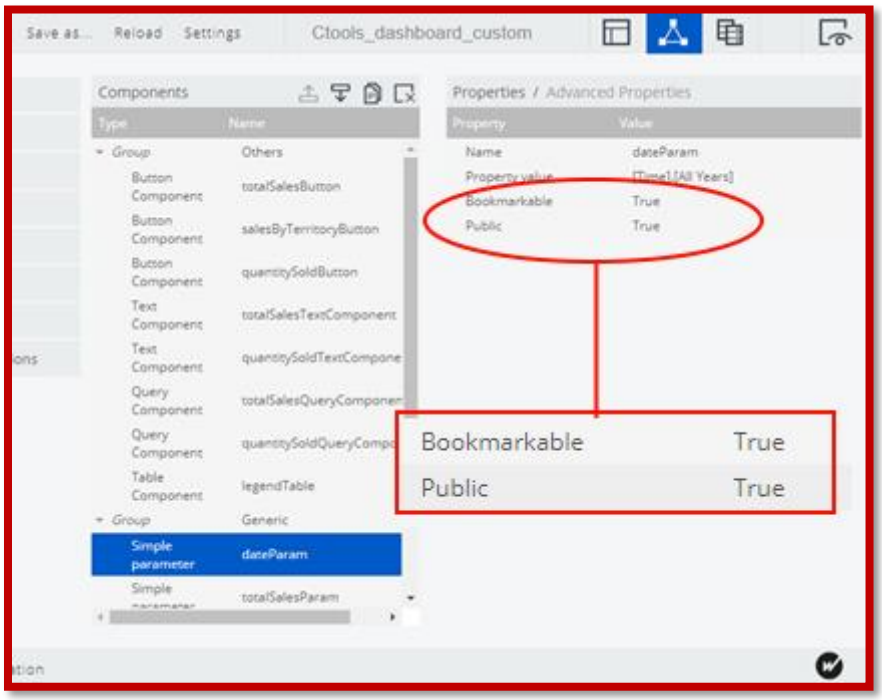
To use the CDE feature you will have to first enable it. Look for more details in the **References** section, Activate CDE.

Examples

```
http://localhost:8080/pentaho/api/repos/:public:Steel Wheels:CTools_dashboard.wcdf/generatedContent
```

Passing parameters

In CTools, parameters are encapsulated in the URL inside a stringified JSON. To make sure you can pass parameters via URL you need to define them as bookmarkable



If you don't know how it looks like, just configure some parameters in your dashboard and mark them as “bookmarkable”. Once you start using the dashboard, if you change the parameters, the JSON line with the parameters will show up in the URL if at least one of them is not using its default value.

Here below an example of the stringified JSON

```
{
   "impl":"client",
   "params":{
      "dateParam":"[Time].[2003]"
   }
}
```

The URL should be structured as follows:

```
/api/repos/<path>/generatedContent?bookmarkState=<stringifiedJSON>
```

Encoding parameters

The stringified JSON should be encoded before pasting it in the browser. You can easily do this using some online tools (https://www.google.de/search?q=url+encode+decode).

Parameter before encoding

```
bookmarkState={"impl":"client","params":{"dateParam":"[Time].[2003]"}}
```

Parameter after encoding

```
bookmarkState=%7B%22impl%22%3A%22client%22%2C%22params%22%3A%7B%22dateParam%22%3A%22%5BTime%5D.%5B2003%5D%22%7D%7D
```

Final URL

```
http://localhost:8080/pentaho/api/repos/:public:Steel%20Wheels:Ctools_dashboard_custom.wcdf/generatedContent?bookmarkState=%7B%22impl%22%3A%22client%22%2C%22params%22%3A%7B%22dateParam%22%3A%22%5BTime%5D.%5B2003%5D%22%7D%7D
```

This example link will not work with the sample out of the box. To test this just open the CTools Dashboard in the Steel Wheels folder to edit. Save it as your new dashboard, and make the parameter dateParam public and bookmarkable.

Create a list of resources in the Pentaho Repository

You can create in your application a combo selection (or a list, or what you prefer) with all resources inside a specific path in the Pentaho Repository, that are of a certain type (an example: display all dashboards in the folder public/Steel Wheels).

With the resulting list, you can create a function onClick that will open the resource in another window or in a specific iframe in your webpage.

Here below an example of code to achieve this.

```
var url = "http://localhost:8080/pentaho/api/repo/files/:public:Steel
Wheels/tree?filter=*.xdash";

$.ajax({
    type: 'GET',
    url: url,
    success: function(json_data, textStatus, jqXHR) {
        _defaultFileListHandler(json_data, textStatus, jqXHR,
containerDivId, path);
    },
    dataType: 'xml'
});

function _defaultFileListHandler(data, textStatus, jqXHR, containerDivId,
rootPath) {
    var newHTML = "<ul>";
    $(data).find('file').each(function() {
        var fileTitle = $(this).find('title').text();
        var filePath = $(this).find('path').text();
        if (filePath != rootPath) { // ignore the root.

            var url = generateViewContentURL(filePath, "view");

            var liHTML = "<li>" +
                "<a href='" + url + "' target='_blank'>" +
                                fileTitle + "</a>" +"</li>";
            newHTML = newHTML + liHTML;
        }
    });

    newHTML = newHTML + "</ul>";

    $("#" + containerDivId).html(newHTML);
}
```

Embedding Self-service Analytics

You can integrate into your application also the self-service experience of Pentaho User Console. In this chapter, we'll introduce the URLs needed to integrate report designer and Analyzer, and then some generic features (like how to save a resource created using the designer).

Interactive Report Designer

New Interactive Report

The following URL will integrate the Report Editor inside an iframe

```
/api/repos/pentaho-interactive-reporting/prpti.new
```

Analyzer

New Analysis Report

The following URL will integrate Analyzer inside an iframe. It will open the schema selector, so the user will be able to choose what schema he wants to base his analysis on

```
/api/repos/xanalyzer/service/selectSchema
```

If you are looking for a more guided experience, and you want to preselect the schema then you can use the following URL.

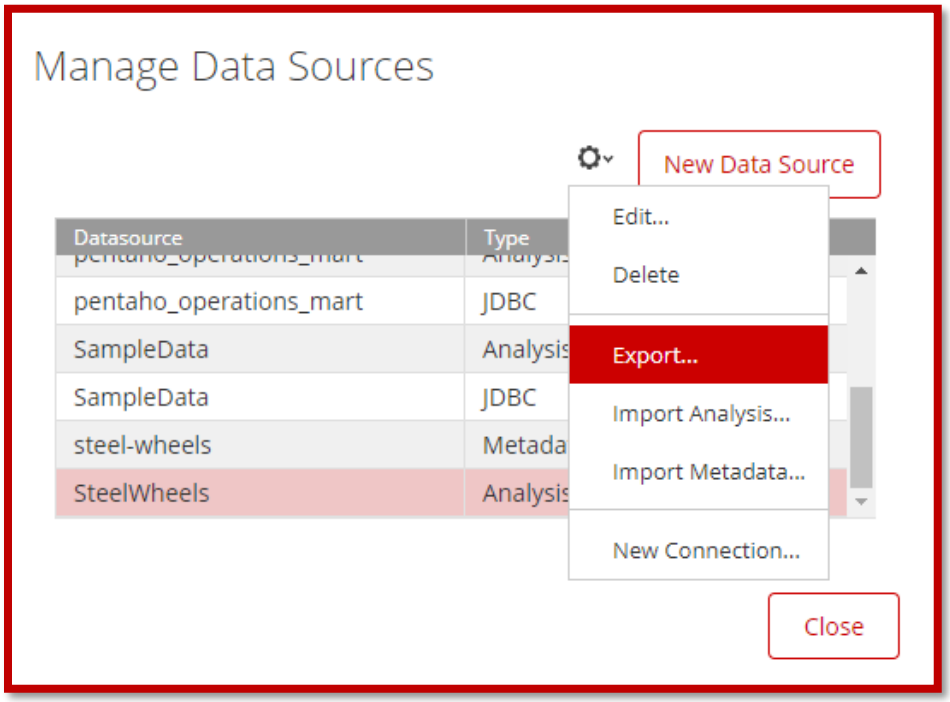
```
/api/repos/xanalyzer/editor?catalog=<Schema>&cube=<CubeName>
```

Find out Schema and CubeName

If you don't know the values of these 2 parameters, you can do the following:

1. Login in Pentaho User Console as Administrator
2. Click on File -> Manage DataSources
3. Select the Analysis Datasource that you want to use in your embedded scenario

4. Click on the gear icon, then Export…



5. Open the exported XML files and you will find the 2 values at the very beginning of the XML definition

```
1  <Schema name="SteelWheels">
2      <Cube name="SteelWheelsSales" cache="true" enabled="true">
3          <Table name="ORDERFACT">
4          </Table>
5          <Dimension foreignKey="CUSTOMERNUMBER" name="Markets">
6              <Hierarchy hasAll="true" allMemberName="All Markets"
               primaryKey="CUSTOMERNUMBER" primaryKeyTable="">
7                  <Table name="CUSTOMER_W_TER">
```

Example

http://localhost:8080/pentaho/api/repos/xanalyzer/editor?catalog=SteelWheels&cube=SteelWheelsSales

To leverage all the functionalities of Analyzer in an embedded scenario continue your reading in the **References** section.

Dashboard Designer

In this section, we'll explain how to integrate the Dashboard Designer self-service experience in your application. It will require some additional steps compared to the other integrations we discussed so far.

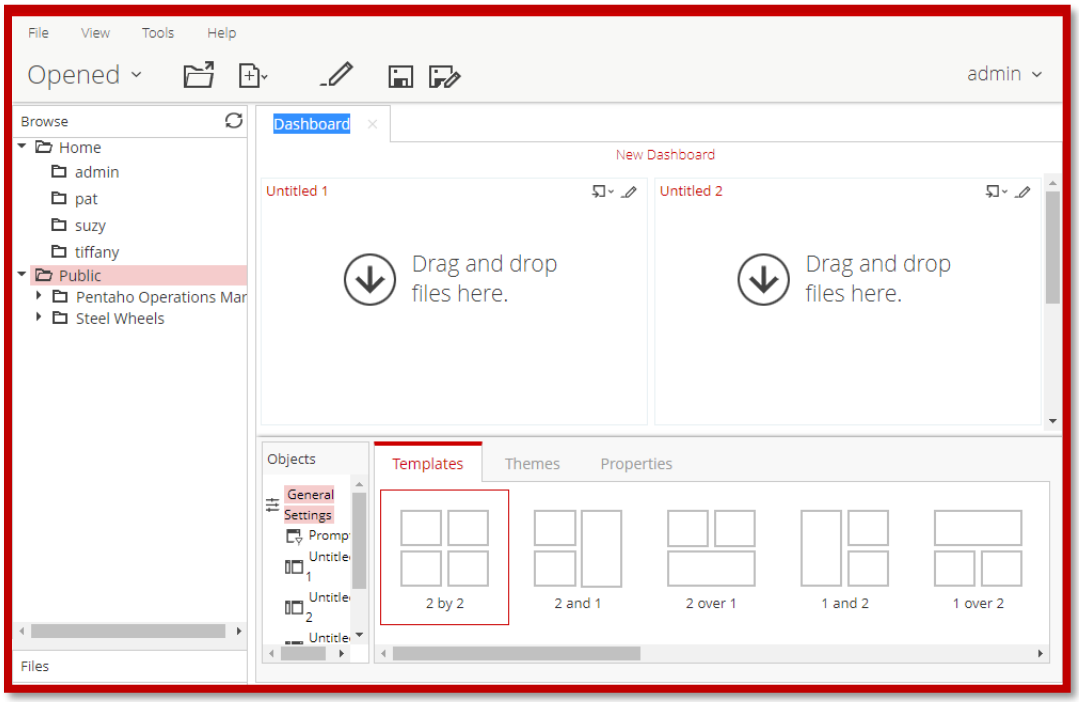
New Dashboard

The following URL will integrate Dashboard Designer inside an iframe.

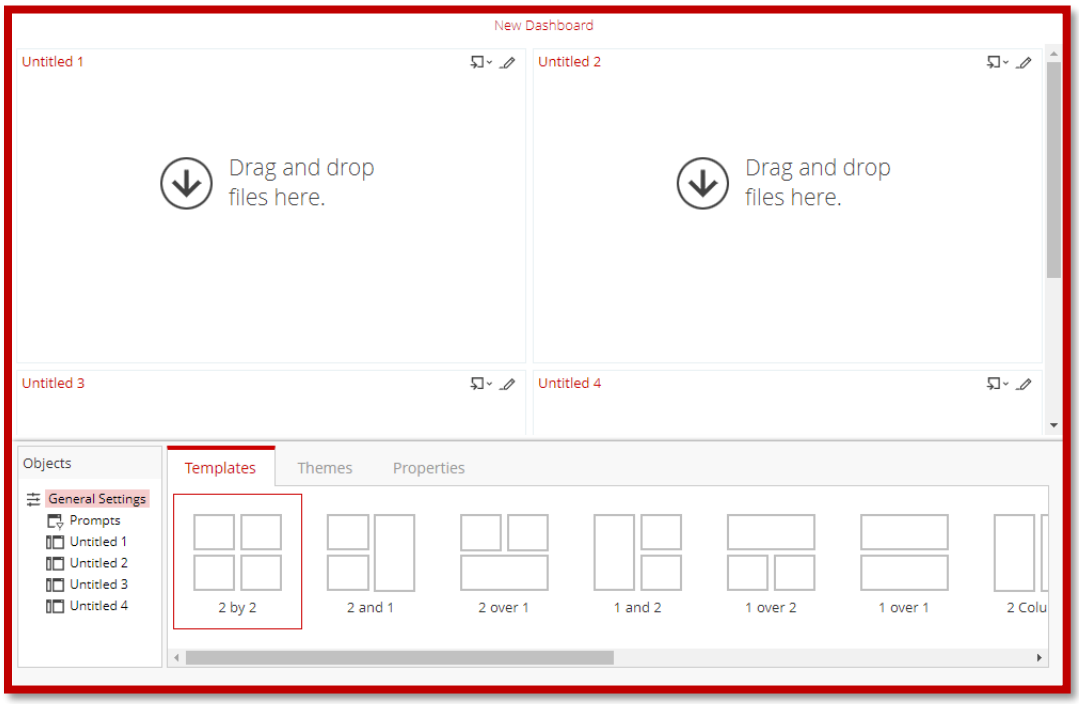
```
/api/repos/dashboards/editor
```

A blank canvas will be opened with the default template selected. You will notice that compared to the Dashboard Editor inside Pentaho User Console you miss something…

Dashboard Designer in Pentaho



Embedded Dashboard Designer



As you probably noticed the left panel is missing… so how can you add widgets to the dashboard? The Insert Content button will not work outside the Pentaho User Console.

Let's get to the solution:

Create a resource selector for Dashboard Designer

The first step is to create a list of resources to select from. This is going to depend on what kind of user experience you want to offer, so it doesn't have to be a list… could be a tree selection, a search box… for sake of simplicity here, you find an example (a simple list).

We'll reuse part of the code shown in the section **Create a list of resources in the Pentaho Repository**

Example code

```
var url = "http://localhost:8080/pentaho/api/repo/files/:public:Steel
Wheels/tree?filter=*.xdash";

$.ajax({
    type: 'GET',
    url: url,
    success: function(json_data, textStatus, jqXHR) {
        _dashboardDesignerResourceListHandler(json_data, textStatus,
jqXHR, containerDivId, path);
    },
    dataType: 'xml'
});


function _dashboardDesignerResourceListHandler(data, textStatus, jqXHR,
containerDivId, rootPath) {

  var cList =
$('<ul>').addClass('widgetSelector').appendTo('#'+containerDivId);

  $(data).find('file').each (function() {
    var fileTitle = $(this).find('title').text();
    var filePath = $(this).find('path').text();
    var fileName = filePath.substr(filePath.lastIndexOf('/') + 1);
    localizedFileName = fileTitle;
    var solution="";
    if (filePath != rootPath) { // ignore the root.

      var li = $('<li/>')
        .addClass('widgetSelector-item')
        .text(fileTitle)
        .appendTo(cList);

        li.click(function(e) {
            addWidgetToDashboard("adminFrame3",solution, filePath,
fileName, localizedFileName);
        });
    }
  });

}
```

Add a resource as widget in the dashboard

In this section, we'll guide you to create a simple Dashboard Designer experience. For this approach to work you will have to first click on the widget canvas on your dashboard layout and then click on the resources on the list created above.

In the Pentaho User Console, you can add resources to the dashboard via a drag&drop. You can achieve the same also in your application by expanding this code and react to drag&drop instead of select&click.
We'll not cover this approach in this guide

The code snippet above is referencing to a function called *addWidgetToDashboard*.
Let's see what is the code needed to add a widget to a dashboard!

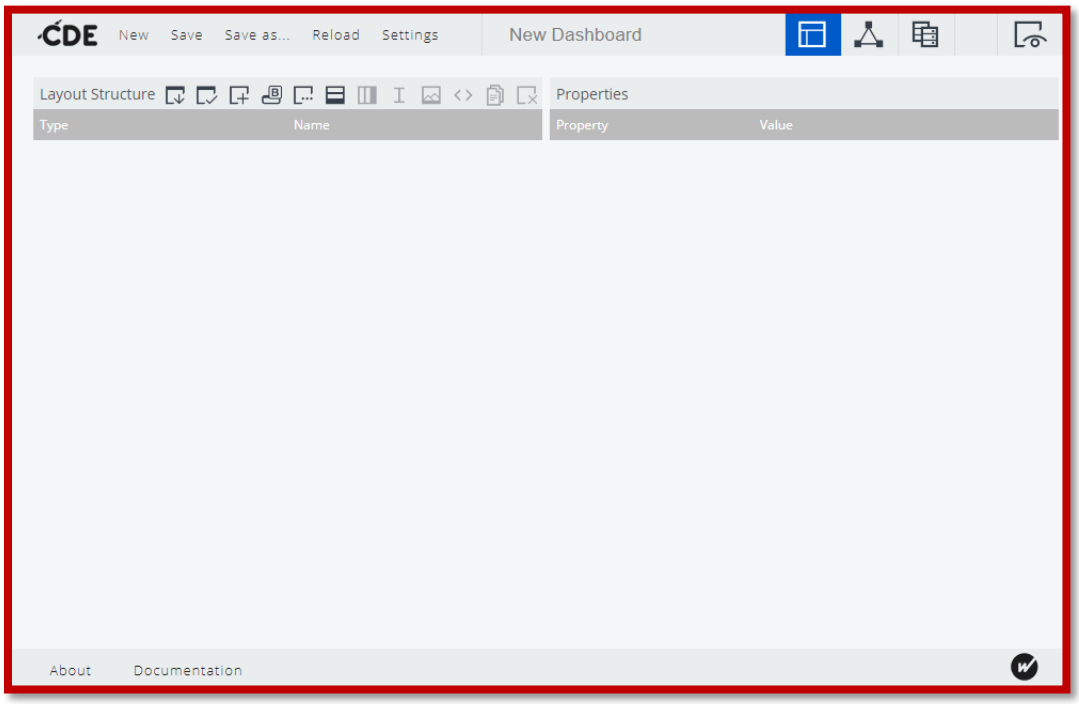
```
function addWidgetToDashboard(iFrameId, solution, path, fileName,
localizedFileName) {
    $("#" + iFrameId)[0].contentWindow
        .SolutionBrowserHelper
            .createComponentForFile(solution, path, fileName,
localizedFileName);
}
```

CDE Dashboard Designer

CDE Dashboards are way more advanced than traditional dashboards, but this comes at a price: they are overly complex for the traditional BI user.
Anyway, if you want to integrate such dashboard editor (maybe only for some advanced users in your application) you can!

```
api/repos/wcdf/new
```

The result will be as shown in the image below.



Additional samples

Add javascript logic to events

If you are integrating a dashboard or a visualization, you may want to fire a specific action when a user double-clicks a slice of a pie (for example)

Here an example of how to achieve this

```
$(window).load(function() {
    $("#demoFrame1")[0].contentWindow.
        pentahoDashboardController.cdfDashboard
            .on('cdf cdf:preExecution', function(e) {
        // I could want to keep a count of all the times
        // my dashboard was visited, for statistical purposes
        console.log('This javascript function is defined in the external
portal. You double clicked on '+e.value)
    });
});
```

Use a helper javascript

All the code shown in this document is useful to understand how Pentaho integration via iFrames work. For a real project, it will probably make sense to encapsulate this logic inside a helper class (or a module if you use require.js or similar libraries).

Here you will find an example of such helper class, alongside some details on how to use it.

This **is not** a production ready code, and it is only meant for **educational purposes**.

Code

The code is available on GitHub:

https://github.com/gianlucanatali/pentaho-embedding/blob/master/pentaho_rest_api_helper.js

https://github.com/gianlucanatali/pentaho-embedding/blob/master/pentaho_helper_usage.js

Example of Usage

```
//--INSTANTIATE THE HELPER
var usernameP = "admin";
var passwordP = "password";
var pentahoSrv = new PentahoRestApis("/pentaho",usernameP,passwordP);

//--INTEGRATE RESOURCES FROM PENTAHO SERVER
//----Dashboard
var dashboardUri = "/public/Steel Wheels/Sales Performance
(dashboard).xdash";
pentahoSrv.loadResourceInDiv(dashboardUri,"view","div_dashboard_1");

//----Report
var reportUri = "/public/Steel Wheels/Top Customers (report).prpt";
pentahoSrv.loadResourceInDiv(reportUri,"view","div_report_1");

//----Analysis Report
var reportUri2 = "/public/Steel Wheels/Widget Library/Analysis
Views/Geomap.xanalyzer";
pentahoSrv.loadResourceInDiv(reportUri2,"view","div_report_2");

//--CREATE LIST OF RESOURCES
var dashboardsPath = "/public/Steel Wheels";
pentahoSrv.loadFileList(dashboardsPath,"*.xdash","resourcesList");
```

```
//--EMBED THE SELF-SERVICE USER EXPERIENCE OF PENTAHO
//----Interactive Report Designer
pentahoSrv.loadResourceInDiv(".prpti","create","div_report_designer");

//----Handle  Save
function saveFile (divId,folderPath) {
    var filename = prompt("Enter file name");
    pentahoSrv.saveResource(divId+'_iframe',filename,folderPath);
}
var folderPath = "/public/Steel Wheels/Widget Library";
$("#save_button_report").click(function() {
    saveFile("div_report_designer",folderPath);

});

//----Analyzer
pentahoSrv.loadResourceInDiv(".xanalyzer","create","div_analyzer");

$("#save_button_analyzer").click(function() {
    saveFile("div_analyzer",folderPath);
});

//----Dashboard Designer
var folderPathDash = "/public/Steel Wheels";
pentahoSrv.loadResourceInDiv(".xdash","create","div_dashboard_des");

function _fileListHandlerForDashboardRes(data, textStatus, jqXHR,
containerDivId, rootPath) {

  var cList =
$('<ul>').addClass('widgetSelectorWs').appendTo('#'+containerDivId);
  $(data).find('file').each (function() {
    var fileTitle = $(this).find('title').text();
    var filePath = $(this).find('path').text();
    var fileName = filePath.substr(filePath.lastIndexOf('/') + 1);
    //var localizedFileName =
($(this).find('localePropertiesMapEntries')[0]).getElementsByTagName("prop
erties")[0].getElementsByTagName("value")[0].innerHTML;
    localizedFileName = fileTitle;
    var solution="";
    if (filePath != rootPath) { // ignore the root.

      var li = $('<li/>')
        .addClass('widgetSelector-item')
        .text(fileTitle)
        .appendTo(cList);

        li.click(function(e) {

pentahoSrv.addWidgetToDashboard("div_dashboard_des_iframe",solution,
filePath, fileName, localizedFileName);
        });
    }
  });
}
```

```
$("#save_button_dashboard").click(function() {
    saveFile("div_dashboard_des",folderPathDash);
});


$("#resourceSelDashDiv").empty();
pentahoSrv.loadFileList(folderPath,"*.xanalyzer%7C*.prpti","resourceSelDas
hDiv",_fileListHandlerForDashboardRes);
```

References

Visit the following links for related information regarding topics discussed in this document.

Pentaho Support documentation:

- Analyzer
  - https://help.pentaho.com/Documentation/8.0/Developer_Center/Analyzer_External_JavaScript_API
  - https://help.pentaho.com/Documentation/8.0/Developer_Center/Analyzer_External_JavaScript_API/User_Interface
- Embedding Pentaho
  - https://help.pentaho.com/Documentation/8.0/Developer_Center/Embed_Pentaho_Server
  - http://mfgaspar.github.io/2016/Embeding-CDF-and-CDE-Dashboards-Part-1/
  - http://pedroalves-bi.blogspot.de/2013/08/embedded-analytics-in-pentaho-with.html
  - http://redmine.webdetails.org/projects/cde/wiki/RequireJS
- Ctools
  - https://help.pentaho.com/Documentation/8.0/Products/CTools/CDE_Quick_Start_Guide
- Activate CDE
  - https://help.pentaho.com/Documentation/8.0/Products/CTools/Activate_CDE

- Older Best Practices (be aware may not be updated to the latest version of Pentaho)
  - https://support.pentaho.com/hc/en-us/articles/206008513-Best-Practices-Test-Driven-Development-Pentaho-Solution